Introduction to KiCad development

Maciej Suminski maciej.suminski@cern.ch

07/09/2016

Before you start

There are a few official documents that one should be familiar with when working with KiCad code. The complete version contains a few more, but the essentials are:

- Coding policy
- User Inteface Guidelines, if you work on the User Interface
- Code documentation

KiCad user interface is built basing on the wxWidgets library, hence it might be beneficial to get to know the wxWidgets basics.

Source tree

This section contains a general description of the KiCad source code directory structure and organization.

Directory	Description
`+-- 3d-viewer`	3D viewer for the board & footprint editor
`+-- CMakeModules`	CMake scripts, mostly used to find 3rd party libraries
`+-- Documentation`	Code documentation
`+-- bitmap2component`	Utility to convert bitmaps to footprint libraries
`+-- bitmaps_png`	Graphics used in the project
`+-- common`	Code shared by eeschema & pcbnew
`    +-- dialogs`	Common dialogs
`    +-- gal`	Graphics Abstraction Layer, new rendering engine
`    +-- geometry`	Geometry libraries used by pcbnew
`    +-- kicad_curl`	cURL library wrapper
`    +-- math`	Generic math library
`    +-- page_layout`	Page layout (worksheet template)
`    +-- tool`	Tool Framework, currently used in pcbnew
`    +-- view`	View Framework, currently used in pcbnew
`    \-- widgets`	Generic widgets
`+-- cvpcb`	Tool for quick footprint assignment
`+-- demos`	Demo projects
`+-- eeschema`	Schematics editor code
`    +-- dialogs`	Dialogs

Directory	Description
`    +-- netlist_exporters`	Netlist export plugins
`    +-- plugins`	Bill of Material generator formatters
`    +-- sim`	Simulator interface
`    \-- widgets`	Additional UI widgets
`+-- gerbview`	Gerber viewer code
`+-- helpers`	Various utilities used to generate certain parts of code (e.g. fonts)
`+-- include`	Headers shared by eeschema & pcbnew (similar contents to common)
`+-- kicad`	Main launcher, project manager
`+-- lib_dxf`	DXF importer/exporter
`+-- new`	Experimental code, not compiled by default
`+-- pagelayout_editor`	Page layout (worksheet template) editor
`+-- patches`	Patches for 3rd party libraries, mostly for OS X
`+-- pcb_calculator`	Utility to compute values used by electronics designers (e.g. transmission lines)
`+-- pcbnew`	Board layout editor
`    +-- autorouter`	Autorouter code, currently available only in the legacy canvas
`    +-- dialogs`	Dialogs
`    +-- exporters`	Board export plugins
`    +-- github`	Github plugin to import footprint libraries
`    +-- import_dxf`	DXF importer code
`    +-- pcad2kicadpcb_plugin`	P-Cad format importer
`    +-- router`	Push and Shove router
`    +-- scripting`	Python scripting support in pcbnew
`    \-- tools`	Tools developed using the Tool Framework
`+-- plugins`	Plugins to handle 3D models import
`+-- polygon`	Polygon handling in pcbnew (e.g. clipping, triangulation)
`+-- potrace`	Bitmap to vector graphics converter
`+-- qa`	Basic unit tests
`+-- resources`	Icons
`+-- scripting`	Python scripting
`+-- scripts`	Extra scripts used by the users (e.g. BOM generation)
`+-- template`	Page layouts (worksheet templates)

Common classes

Even though there are hundreds of classes in KiCad, a few of them are used so frequently that almost surely you will work with them:

General

- `DLIST`

Doubly-linked list, frequently used for storing items in schematic or board objects.

- `EDA_ITEM`

The base class for any kind of item that can be placed on a schematic sheet or a board.

- `EDA_BASE_FRAME`

The base class for all windows. You can easily find a class dedicated to the part you are working with (e.g. footprint editor is `FOOTPRINT_EDIT_FRAME`).

- `KIFACE` & `KIWAY`

These classes are used for inter tool communication, when KiCad is launched in a project mode (i.e. kicad executable and a project loaded). This is how e.g. cross probing between eeschema & pcbnew is done.

- `PROJECT`

Stores project related settings.

- `UNDO_REDO_CONTAINER` & `PICKED_ITEMS_LIST`

These two classes are used to handle undo/redo buffer. For most of the time they are used by `SaveCopyInUndoList()` methods.

eeschema

- `SCH_ITEM`

The base class for all items possibly placed on a schematic sheet.

- `SCH_SCREEN`

`SCH_SCREEN` is currently the main data model in eeschema, counterpart of `BOARD` class in pcbnew. It corresponds to a single schematic sheet (note that eeschema supports hierarchical schematics, so you may have more than one `SCH_SCREEN` per file).

pcbnew

- `BOARD_ITEM`

`BOARD_ITEM` inherits from `EDA_ITEM` and is used as the base class for items placed on a board. There is also `BOARD` class which represents the edited PCB.

- `PLUGIN`

The plugin interface for importing footprint libraries in pcbnew.

- `TOOL_INTERACTIVE`

The base class for the interactive tools in the new pcbnew canvases.

- `TOOL_MANAGER`

TOOL_MANAGER is the way for tools written using the Tool Framework to get reach the external world: view, data model, other tools.

- VIEW

VIEW is the class responsible for managing items to be drawn in the new canvases.